Efficient Techniques for Tree Similarity Queries

Nikolaus Augsten

Database Research Group
Department of Computer Sciences
University of Salzburg, Austria

July 6, 2017

Austrian Computer Science Day 2017 / IMAGINE 17

1 Partially funded by Austrian Science Fund (FWF), FFTED Project P 29859
Trees Are Everywhere

- data formats (XML, JSON), directory hierarchies
- ERP: human resources, enterprise assets, bills of material
- NLP: syntax trees of natural languages
- software evolution: source code
- bioinformatics: RNA secondary structure, carbohydrates, neuronal morphology, phylogenetic trees
- image recognition: gesture and shape recognition
- astrophysics: ancestry trees of galaxies (merger trees)

Source: www.inspirehep.net

(19M trees, total of 750M nodes)
### Similarity Join on Trees

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_1$</td>
<td></td>
<td>$T_1'$</td>
</tr>
<tr>
<td></td>
<td>$T_2$</td>
<td></td>
<td>$T_2'$</td>
</tr>
<tr>
<td></td>
<td>$T_3$</td>
<td></td>
<td>$T_3'$</td>
</tr>
<tr>
<td>tid</td>
<td>tree</td>
<td></td>
<td>tree</td>
</tr>
<tr>
<td></td>
<td>$T_1$</td>
<td></td>
<td>$T_1'$</td>
</tr>
<tr>
<td></td>
<td>$T_2$</td>
<td></td>
<td>$T_2'$</td>
</tr>
<tr>
<td></td>
<td>$T_3$</td>
<td></td>
<td>$T_3'$</td>
</tr>
</tbody>
</table>

- $T_1$: $x \quad y \quad w \quad v \quad z$
- $T_2$: $a \quad b \quad c \quad b$
- $T_3$: $a \quad b \quad h \quad e$
- $T_1'$: $a \quad b \quad c \quad d \quad e$
- $T_2'$: $d \quad a \quad h \quad i$
- $T_3'$: $x \quad y \quad w \quad v \quad z$

Nikolaus Augsten (Univ. of Salzburg)
Similarity Join on Trees

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td></td>
<td>b</td>
</tr>
<tr>
<td>T1</td>
<td>w</td>
<td></td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>v</td>
<td></td>
<td>d</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td></td>
<td>e</td>
</tr>
</tbody>
</table>

|     | 5          |     | 2          |
|     | 6          |     | 5          |

|     | 4          |     | a          |
| T2  | b          |     | b          |
|     | c          |     | h          |

|     | 3          |     | 1          |
|     | 5          |     | x          |
|     | 5          |     | w          |
|     | 5          |     | w          |
|     | 5          |     | z          |

|     | T1'        |     |
|     |            |     |

|     | T2'        |     |
|     |            |     |

|     | T3'        |     |
|     |            |     |
Similarity Join on Trees

\[ F \]

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>y w</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>v z</td>
</tr>
</tbody>
</table>

\[ T_1 \]

\[ F' \]

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>b c d</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>e</td>
</tr>
</tbody>
</table>

\[ T_1' \]

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>a h i</td>
</tr>
</tbody>
</table>

\[ T_2 \]

\[ T_2' \]

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>y w</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>w z</td>
</tr>
</tbody>
</table>

\[ T_3 \]

\[ T_3' \]

<table>
<thead>
<tr>
<th>tid</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
</tr>
<tr>
<td></td>
<td>b</td>
</tr>
</tbody>
</table>

\[ F' \]

\[ \text{threshold}=2 \]
Outline

1. How similar are two trees?
2. From trees to integer sets
3. Avoiding the nested loop join
1. How similar are two trees?

2. From trees to integer sets

3. Avoiding the nested loop join
Tree Edit Distance (TED)

- minimum number of edit operations to transform one tree into another
How similar are two trees?

Tree Edit Distance (TED)

- minimum number of edit operations to transform one tree into another

TED = 3

Nikolaus Augsten (Univ. of Salzburg)
Efficient Tree Similarity Queries
July 6, 2017 6 / 24
### State of the Art in TED

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Space</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang&amp;Shasha 1989</td>
<td>$O(n^4)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2 \log^2 n)$ runtime for balanced trees</td>
</tr>
<tr>
<td>Demaine et al. 2009</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
<td>worst case is frequent</td>
</tr>
</tbody>
</table>

**Zhang 0.5 sec**
**Demaine 30.9 sec** (≈ 1000 nodes)

**Zhang 247.6 sec**
**Demaine 26.0 sec** (≈ 1000 nodes)
## State of the Art in TED

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Space</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang &amp; Shasha 1989</td>
<td>$O(n^4)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2 \log^2 n)$ runtime for balanced trees</td>
</tr>
<tr>
<td>Demaine et al. 2009</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
<td>worst case is frequent</td>
</tr>
</tbody>
</table>

Zhang: 0.5 sec
Demaine: 30.9 sec

($\approx 1000$ nodes)
## State of the Art in TED

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Space</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang &amp; Shasha</td>
<td>$O(n^4)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2 \log^2 n)$ runtime for balanced trees</td>
</tr>
<tr>
<td>Demaine et al.</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
<td>worst case is frequent</td>
</tr>
</tbody>
</table>

### Time and Space Complexities

- **Zhang**: 0.5 sec (≈ 1000 nodes)
- **Demaine**: 30.9 sec (≈ 1000 nodes)
- **Zhang**: 247.6 sec (≈ 1000 nodes)
- **Demaine**: 26.0 sec (≈ 1000 nodes)
State of the Art in TED

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Space</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang&amp;Shasha 1989</td>
<td>$O(n^4)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2 \log^2 n)$ runtime for balanced trees</td>
</tr>
<tr>
<td>Demaine et al. 2009</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
<td>worst case is frequent</td>
</tr>
</tbody>
</table>

Problem: Hard-coded decomposition strategies
How similar are two trees?

Robust Tree Edit Distance Algorithm (RTED)

strategy computation
$O(n^2)$ time and space

optimal strategy

distance computation
$O(n^3)$ time and $O(n^2)$ space

TED
Robust Tree Edit Distance Algorithm (RTED)

How similar are two trees?

RTED: fastest known algorithm for TED.

Algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Year</th>
<th>Time</th>
<th>Space</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang &amp; Shasha</td>
<td>1989</td>
<td>$O(n^4)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2 \log^2 n)$ runtime for balanced trees</td>
</tr>
<tr>
<td>Demaine et al.</td>
<td>2009</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
<td>worst case is frequent</td>
</tr>
<tr>
<td>Pawlik &amp; Augsten (RTED)</td>
<td>2011</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
<td>optimal strategy</td>
</tr>
</tbody>
</table>
Performance Evaluation

Runtime

- measure runtime (seconds)
- vary tree shape
- tree size $\approx$ 1000 nodes

RTED: fastest known algorithm for TED.
Outline

1. How similar are two trees?
2. From trees to integer sets
3. Avoiding the nested loop join
Similarity Join on Trees

\[ F \]
\begin{align*}
\text{tid} & \quad \text{tree} \\
T_1 & \quad \begin{array}{c}
\begin{array}{c}
\times \\
/ \quad \backslash
\end{array} \\
y \quad w \\
/ \quad \backslash \\
v \quad z
\end{array} \\
T_2 & \quad \begin{array}{c}
\begin{array}{c}
a \\
/ \quad \backslash
\end{array} \\
b \quad c \quad b
\end{array} \\
T_3 & \quad \begin{array}{c}
\begin{array}{c}
a \\
/ \quad \backslash
\end{array} \\
e \quad b \quad h
\end{array}
\end{align*}

\[ F' \]
\begin{align*}
\text{tree} & \quad \text{tid} \\
T_1' & \quad \begin{array}{c}
\begin{array}{c}
a \\
/ \quad \backslash
\end{array} \\
b \quad c \quad d \quad e
\end{array} \\
T_2' & \quad \begin{array}{c}
\begin{array}{c}
d \\
/ \quad \backslash
\end{array} \\
a \quad h \quad i
\end{array} \\
T_3' & \quad \begin{array}{c}
\begin{array}{c}
x \\
/ \quad \backslash
\end{array} \\
y \quad w \\
/ \quad \backslash \\
w \quad z
\end{array}
\end{align*}

Nested loop join (NLJ):
1. compute distance between all pairs of trees
2. return trees within threshold

Very expensive:
- \( N^2 \) distance computations!
Nested loop join (NLJ):
1. compute distance between all pairs of trees
**Similarity Join on Trees**

- **Nested loop join (NLJ):**
  1. compute distance between all pairs of trees
  2. return trees within threshold

### Table

<table>
<thead>
<tr>
<th>tid</th>
<th>F</th>
<th>F'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tree</td>
<td>tree</td>
</tr>
<tr>
<td>tid</td>
<td>tree</td>
<td>tree</td>
</tr>
<tr>
<td></td>
<td>x / \ y w / \ v z</td>
<td>a / \ b c d / \ e</td>
</tr>
<tr>
<td></td>
<td>a / \ b c b</td>
<td>d / \ a h i</td>
</tr>
<tr>
<td></td>
<td>a / \ e b h</td>
<td>1 / \ x w / \ w z</td>
</tr>
</tbody>
</table>
Similarity Join on Trees

From trees to integer sets

Similarity Join on Trees

Threshold = 2

Nested loop join (NLJ):
1. compute distance between all pairs of trees
2. return trees within threshold

Very expensive: $N^2$ distance computations!
### Sorting: Errors destroy sort order

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frieda</td>
<td>Salzburg</td>
</tr>
<tr>
<td>Frodo</td>
<td>Auenland</td>
</tr>
<tr>
<td>Maria</td>
<td>Wien</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frieda</td>
<td>Salzburg</td>
</tr>
<tr>
<td>Maria</td>
<td>Wien</td>
</tr>
<tr>
<td>Vrodo</td>
<td>Auenland</td>
</tr>
</tbody>
</table>
### From trees to integer sets

#### Standard Join Techniques Fail

- **Sorting:** Errors destroy sort order

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frieda</td>
<td>Salzburg</td>
</tr>
<tr>
<td>Frodo</td>
<td>Auenland</td>
</tr>
<tr>
<td>Maria</td>
<td>Wien</td>
</tr>
</tbody>
</table>

- **Hashing:** Errors change hash value

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

---

Nikolaus Augsten (Univ. of Salzburg)  Efficiency Tree Similarity Queries  July 6, 2017
pq-Grams: From Trees to Sets

- **Split trees** into pq-Grams (small subtrees of fixed shape)

  ![Diagram of trees and pq-Grams]

- Hash pq-grams to integers

- **Intuition:** Similar trees many common pq-grams
Efficiency

- pq-gram runtime: $O(n \log n)$
- edit distance runtime: $O(n^3)$

Filter for edit distance

- good approximation
- guarantees lower bound (fanout weighting of operations)

Tree similarity simplified to set overlap.
Outline

1. How similar are two trees?
2. From trees to integer sets
3. Avoiding the nested loop join
Avoiding the nested loop join

Nested Loop Join for Sets – A Small Experiment

- **Dataset**: title and author of \( n = 873k \) publications (DBLP)
  - avg. 15 words (min: 2, max: 289)
  - 409k different words, Zipfian frequency distribution
Avoiding the nested loop join

Nested Loop Join for Sets – A Small Experiment

- **Dataset**: title and author of $n = 873k$ publications (DBLP)
  - avg. 15 words (min: 2, max: 289)
  - 409k different words, Zipfian frequency distribution

- **Goal**: find all similar pairs (e.g. 90% overlap)

Similarity check is very fast: 6 ns ≈ 15 CPU cycles

Nested loop join: too many comparisons! (approx. 381 billion)

Overlap

<table>
<thead>
<tr>
<th>Runtime [s]</th>
<th>0.95</th>
<th>0.9</th>
<th>0.8</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2201</td>
<td>3136</td>
<td>4280</td>
<td>7770</td>
</tr>
</tbody>
</table>
Nested Loop Join for Sets – A Small Experiment

- **Dataset:** title and author of \( n = 873k \) publications (DBLP)
  - avg. 15 words (min: 2, max: 289)
  - 409k different words, Zipfian frequency distribution

- **Goal:** find all similar pairs (e.g. 90% overlap)

- **Similarity check is very fast:** 6 ns \( \approx \) 15 CPU cycles

---

Nikolaus Augsten (Univ. of Salzburg)  Efficient Tree Similarity Queries  July 6, 2017  16 / 24
**Dataset:** title and author of \( n = 873k \) publications (DBLP)
- avg. 15 words (min: 2, max: 289)
- 409k different words, Zipfian frequency distribution

**Goal:** find all similar pairs (e.g. 90% overlap)

**Similarity check is very fast:** 6 ns \( \approx \) 15 CPU cycles

**Nested loop join:** too many comparisons! (approx. 381 billion)

<table>
<thead>
<tr>
<th>Overlap</th>
<th>0.95</th>
<th>0.9</th>
<th>0.8</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime [s]</td>
<td>2201</td>
<td>3136</td>
<td>4280</td>
<td>7770</td>
</tr>
</tbody>
</table>

**NLJ does not even scale for very fast checks.**
Prefix filter¹: only look at small part of the set
  - sort sets and cut after prefix
  - no overlap in prefix \(\Rightarrow\) similarity threshold not reachable

\[
\begin{array}{c}
\text{r: a c e f g} \\
\text{s: b d e f g}
\end{array}
\]

| r \cap s | \geq 4

Prefix Index Join

collection $S$
prefix index

lookup collection of probing sets $R$

prefix index $N$
Prefix Index Join

collection S
prefix index

lookup

collection of
probing sets R

a
s_1
s_2
s_3
s_4
s_5
c

a
r_6
c
r_7
### Comparison to Nested Loop Join

<table>
<thead>
<tr>
<th>Overlap</th>
<th>Nested Loop Join</th>
<th>Prefix Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>2201 s</td>
<td>0.26 s</td>
</tr>
<tr>
<td>0.90</td>
<td>3136 s</td>
<td>0.44 s</td>
</tr>
<tr>
<td>0.80</td>
<td>4280 s</td>
<td>1.54 s</td>
</tr>
<tr>
<td>0.70</td>
<td>7770 s</td>
<td>4.93 s</td>
</tr>
</tbody>
</table>

Prefix Join 1500 to 8000 times faster than nested loop join!
Summary

- **Set similarity join**
  - RTED: efficient tree edit distance
  - pq-grams: from trees to sets
  - prefix: avoiding nested loop join

- **Publications**
  - Very Large Database Conference (VLDB 2005, 2011, 2016)
  - Elsevier Information Systems (Inf. Syst. 2016)

- **Algorithms and technology matter**
  - efficient solutions orders of magnitude faster
Conclusion and Outlook

DACH publications in top-3 DBS outlets: only 1.5% from Austria

Master Data Science in Salzburg:
data analysis + applications + technology

Efficient technology is key enabler for innovation in big data.

Nikolaus Augsten (Univ. of Salzburg)
Efficient Tree Similarity Queries
July 6, 2017 21 / 24
Efficient technology is key enabler for innovation in big data.
Conclusion and Outlook

- DACH publications in top-3 DBS outlets: only 1.5% from Austria

Efficient technology is key enabler for innovation in big data.
DACH publications in top-3 DBS outlets: only 1.5% from Austria

Master Data Science in Salzburg:
- data analysis + applications + technology

Efficient technology is key enabler for innovation in big data.
DACH publications in top-3 DBS outlets: only 1.5% from Austria

Master Data Science in Salzburg:
  data analysis + applications + technology

Invitation: VLDB 2017 in Munich, EDBT 2018 in Vienna!

Efficient technology is key enabler for innovation in big data.
Thanks for your attention!


Mateusz Pawlik and Nikolaus Augsten.  
RTED: A robust algorithm for the tree edit distance.  
ISSN 2150-8097.

Mateusz Pawlik and Nikolaus Augsten.  
Efficient computation of the tree edit distance.  
*ACM Transactions on Database Systems (TODS)*, 40(1):3:1–3:40,  
March 2015.  
ISSN 0362-5915.

Mateusz Pawlik and Nikolaus Augsten.  
Tree edit distance: Robust and memory-efficient.  
ISSN 0306-4379.